BLOCKSTAND

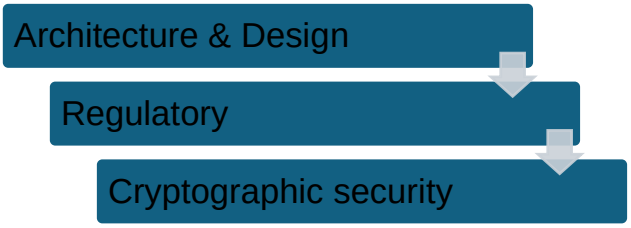
Standardizing On-chain Attestation for EU Web3 Passport

Abstract

European digital identity is at a crossroads, balancing cross-border interoperability with strict privacy mandates. A "European Web3 Passport" would empower citizens to prove identity attributes (age, citizenship, etc.) when accessing decentralized services across EU Member States without surrendering excessive personal data. Achieving this scope requires marrying blockchain based services networks with zero-knowledge (ZK) attestations, which allow proving facts about data without revealing the data itself, and Trusted Execution Environments (TEEs) that ensure sensitive computations occur securely.

The European Blockchain Services Infrastructure (EBSI), a public permissioned EU-wide ledger, offers a ready backbone that can be empowered as on chain registry and aligned with EU regulations. By leveraging EBSI for on-chain trust anchoring, and ZK proofs and TEEs for privacy and security, we can design an architecture for a Europe-wide digital passport that is privacy-preserving, interoperable, and compliant with eIDAS 2.0 and GDPR.

This work explores a way to standardize design and architecture for implementing Zero Knowledge on-chain attestations to the intent of creating a European Web3 passport and how TEEs fortify their security, discusses regulatory compliance in such a system, reviews key cryptographic primitives enabling the solution.



Architectural & Design Aspects

ZK Attestations Structure	ZKP & TEE
	Trust Framework
	Storage

1. Structure of ZK On-Chain Attestations

In a digital identity system, an attestation is a claim about an entity (e.g. "Person X is over 18") made by a trusted issuer. Traditional Web3 attestations often treat identity as an aggregate of attestations tied to an identifier into a presentation. For a European Web3 Passport, the claims are encoded in a privacy-preserving way using ZK on-chain attestations. This means the existence or validity of an attestation is anchored on-chain (for integrity and interoperability), while the content can be proven via zero-knowledge proofs.

Rather than publishing personal data, the blockchain stores commitments cryptographic evidence of identity data. A common design is to use a Merkle tree: each user aggregates their identity attributes or credentials as leaves in a Merkle tree, whose root hash serves as a commitment to all attestations. This root, which is often called an Identity State, is published on-chain, typically in a smart contract or DID document, and updates whenever the user adds or revokes an attestation. For example, the

Iden3/Polygon ID protocol uses a Merkle root as the user's identity state, which is stored on-chain to enable thrustless verification[1]. In such a design, a user can receive multiple attestations (e.g. a government-issued citizenship credential, an age verification, a diploma), store commitments to each in their Merkle tree, and only the root is posted to the ledger. The ledger thus holds a tamper-proof commitment to the user's data without revealing it. An alternative, or complementary design option is a shared on-chain attestation registry contract[2] where issuers publish attestations (still as hashes or encrypted blobs) to a common ledger.

Ideally, credentials would be issued directly in ZK-friendly formats—such as Merkleized credentials or standardized attribute-level commitments allowing efficient and privacy-preserving on-chain storage of minimal data (roots or commitments). The EBSI-hosted Trusted Issuers and Attestation Registries can then store only these optimized ZK-friendly commitments, significantly simplifying verification and enhancing data minimization compliance.

Each user (and issuer) is identified by a DID, such as did:ebsi for official EU identities[3]. Attestations are linked to these on-chain identifiers. For example, a government issuer's DID sign an attestation for a holder's DID. The on-chain registry might record something like: (holder DID) – has attestation of type X – issued by (issuer DID) – commitment Y. Because EBSI's DID method aligns with EU regulations, using did:ebsi for issuers and linking to the European Trusted Issuers Registry ensures that any verifier in the EU can cryptographically trust the attestation's origin (the issuer's public key is known and authorized) without needing to contact the issuer directly each time.

With the commitments in place on-chain, the user can generate a ZK proof off-chain to demonstrate a certain attribute without revealing the underlying data. For instance, the user can prove "I possess an attestation issued by DID:Government that my age >= 18", by proving knowledge of a valid attestation in their Merkle tree that contains a birthdate satisfying the age check. The on-chain data (like the latest Merkle root committed for that user, or a registry entry) is used by the verifier contract as a reference. This way, the heavy personal data (birthdate, name, passport number) never leaves the user's wallet; only a proof and perhaps a nullifier (unique one-time token, discussed later) are revealed.

When new attestations are added or revoked, the user (or issuer) updates the on-chain commitment. This mechanism preserves integrity: only the holder (with their private keys) can update their state, and they must prove the update is valid. Thus, the on-chain structure of ZK attestations involves two components:

(1) Storage of commitments (identity root, revocation lists, etc.),

(2) Verification logic (smart contracts that verify ZK proofs against those commitments).

2. Trusted Execution Environments (TEE) in Secure Verification

Trusted Execution Environments provide a hardware-isolated sandbox for computations, offering security properties like memory encryption and protected execution. In this context, TEEs (such as Intel

[1] Iden3 protocol specs - Iden3 Documentation) (Configuration | Privado ID Documentation

[2] ERC-7812: ZK Identity Registry

[3] Verifiable Credentials and Decentralised Identifiers: Technical Landscape

SGX, ARM TrustZone, or TPM-backed secure enclaves) can be leveraged to enhance the security of both issuance and verification of ZK attestations:

- **Secure Credential Issuance:** Consider a government issuer creating a ZK-friendly credential (e.g. a signed hash of your passport data). Using a TEE, the issuer can ensure the passport data is handled only within an enclave that signs the attestation and immediately discards the plaintext. The TEE's remote attestation capability can prove to the outside world (or to an audit log) that the code running was the approved issuance logic and that, for example, no copy of the user's personal data was kept[4]. This adds a layer of trust: even if the issuing server is compromised, the private signing key and data processing remain secure inside the enclave Only the attestation result (a signed credential or commitment) leaves the enclave. In scenarios where biometric data or other highly sensitive attributes are processed (e.g., iris scans, fingerprints embedded into e-passports, or next-generation credentials), TEEs can secure the handling of this sensitive biometric data. For interoperable cross-border identity verification, TEEs can help governments prove to external partners and citizens from other EU countries that credentials are issued securely and transparently.

- **Privacy-Preserving Proof Generation:** On the user side, generating certain ZK proofs can be computationally intensive or require handling sensitive data (like biometric or document data). TEEs can assist users who may not have capable devices. For instance, if a user's smartphone cannot efficiently compute a zk-SNARK proving the validity of their biometric passport the user could securely offload this task to a cloud TEE service. The raw passport data would be sent encrypted to a server enclave; the enclave (which can be attested to be running the correct ZK prover code) generates the proof and returns it. Crucially, the enclave ensures the data is not leaked in the process. Because the TEE is attested, the verifier (or the user) can be confident that only the intended computation was performed, and no personal info was exposed beyond what the ZK proof reveals. This pattern – a TEE-assisted prover – adds some dependency on hardware trust, but can greatly improve usability while still keeping data confidential.

- **Key Management and Device Security**: The user's private keys (for their DID, and any secret values used in ZK proofs) should be securely stored. Modern smartphones include secure enclaves (like Android StrongBox or Apple Secure Enclave) that can act as TEEs to store keys and even perform cryptographic operations. A European Digital Identity Wallet could utilize these hardware modules to protect the user's identity keys, ensuring that even if the phone's OS is malware-infected, the keys aren't stolen. For example, signing a new identity state or generating a selective disclosure proof could involve the secure enclave for the signing step. TEEs thus uphold credential integrity and mitigate the risk of impersonation.

It's important to note that TEEs do require trust in the hardware provider and can be vulnerable to certain side-channel attacks. However, when combined with ZK proofs, they address complementary threats: ZK proofs remove the need to reveal data to verifiers (protecting privacy even if the verifier is malicious), while TEEs remove the need to trust the execution environment (protecting against malware or rogue issuers). In fact, proposals have been made to use SGX enclaves to hold the secrets that derive

[4] Lumoz unveils TEE+ZK multi-proof for on-chain AI Agent | CoinMarketCap

nullifiers in identity systems, so that not even the server operating the enclave can link users to their nullifiers[5].

In summary, TEEs bolster the security of a ZK attestation system by ensuring that verification and proof-generation steps are untampered, and by enabling more complex operations (like verifying ICAO passport chips or X.509 signatures) to be done in a confined space when they are too expensive to do in zero-knowledge on user devices.

EU-Wide Interoperability Considerations

A potential Web3 Passport to be useful across Europe, it should be interoperable by design, technologically and legally, so that an attestation issued in one member state can be recognized in another. The European Blockchain Services Infrastructure (EBSI) is central to this interoperability. It is a pan-European distributed ledger, governed by Europeum EDIC, intended to share trusted data across borders. EBSI operates nodes in multiple countries a permissioned DLT based on a Proof of Authority by Member States organizations.

Key interoperability design patterns may include:

- **Standardized Identity Data Models:** W3C Verifiable Credentials (VC) and Decentralized Identifiers (DID) standards which eIDAS 2.0 and the upcoming European Digital Identity (EUDI) Wallet may embrace to offer interoperable solutions with web3 services. A VC is a generic container for attestations; by following the VC Data Model, our Web3 Passport credentials can be understood by any conformant wallet or verifier. EBSI itself has defined profiles for verifiable diplomas, social security credentials, this should also include the support for selective disclosure and ZK proofs as privacy-enhancing techniques. For instance, the claim “over 18” could be modeled as a VC issued by a national ID authority under a standard schema, which all verifiers across the EU recognize.

- **DID Methods and Resolution:** Each entity’s DID must be resolvable across borders. The did:ebsi method was specifically created for European digital identities and is supported by EBSI’s DID registry. By using did:ebsi for government issuers and perhaps for citizens (did:ebsi may support legal entities and could support natural persons via the EBSI wallet, any verifier could look up the DID document to obtain the issuer’s public keys and service endpoints. Interoperability is thus baked in; a verifier in Italy or Spain or other countries can fetch the DID document of, say, the Estonian government’s issuer DID to verify a credential it issued. EBSI ensures these DID records are governed and trusted, in fact only authorized bodies can register certain DIDs, preventing spoofing[6].

- **European Trust Framework Integration:** eIDAS 2.0 is establishing a European Digital Identity Framework where each member state will recognize the others’ “electronic attestations of attributes” and digital identities. In practice, this means if France issues a digital driving license credential, a verifier in Germany should accept it given common rules. Our design supports this by having common verification infrastructure. If both are connected to EBSI, they rely on the same smart contracts and registries

[5] Privacy preserving nullifiers for proof of identity applications - zk-s[nt]arks - Ethereum Research

[6] Verifiable Credentials and Decentralised Identifiers: Technical Landscape) (Request and verify Verifiable Credentials | EBSI hub

- **Cross-Chain and Legacy Integration:** While EBSI might be the primary ledger, the system should not be siloed. The attestations should be usable in the broader Web3 ecosystem to maximize utility. For instance, a user might want to prove their EU Web3 Passport traits on an Ethereum dApp or a social media platform. Projects like, Ethereum Foundation Open Passport, Privado ID, have already demonstrated cross-chain identity interoperability, allowing proofs generated from one network to be verified on another via interoperability protocols[7]. Potentially, It may be possible to leverage existing solutions to obtains a proof against EBSI data with relay contract on Ethereum which knows how to verify an EBSI-origin proof, or check a signature from an EBSI oracle.
- **Interoperable Storage & Revocation:** All member states should be able access consistent revocation information and latest data. By using a single source (or a federated but synchronized source) for the Merkle roots or credential status, it may be possible to reduce data fragmentation. For example, if a passport is revoked (perhaps the citizen loses it or it expires), that revocation should be reflected so any verifier can detect it. EBSI can host a revocation registry that issuers update. Since our design uses commitments, revocation might mean removing or marking a leaf in a Merkle tree. The on-chain state (Merkle root) would update, and verifiers will require proofs against the latest root. EBSI's consensus ensures all verifiers see the updated root approximately simultaneously, achieving consistency across borders.

In summary, interoperability is achieved by building on common standards and infrastructure: use EBSI for the ledger and trust logic, use W3C VC/DID for data formats, and align with eIDAS's legal framework for cross-recognition.

3. Storage Mechanisms: On-Chain zk Commitments vs Off-Chain Merkle Proofs

A crucial design decision is what data to store on-chain versus off-chain. A guiding design principle should be orienting privacy and efficiency: sensitive personal data stays off-chain; only cryptographic assurances go on-chain. The implementation via on-chain zkCommitments and off-chain proofs should achieve this:

- **On-Chain Commitments:** These are one-way references to data (usually hashes). The Merkle root for instance is one such commitment. Another example is a simple hash of a credential (e.g. hash of a passport's serial number and expiry date). Storing commitments on-chain provides integrity and a point of reference for proofs. For instance, the hash of the user's passport data is inserted into an on-chain Merkle tree. The Merkle tree's root is stored on-chain (inside a contract), effectively committing to all registered passports. Each user's inclusion can later be proven. This way, the expensive operation of checking a passport's authenticity can be done once at registration, and thereafter a lightweight inclusion proof is enough. A potential user story may be like "This person has a valid national ID" may be committed on-chain as a single bit or hash, allowing anyone to verify a proof of inclusion or validity without re-checking the original document.

  While explicit on-chain authenticity checks may indeed seem redundant for trusted government-issued credentials, placing cryptographic commitments on-chain serves broader

[7] Privado ID Universal Verifier, Ethereum Foundation, Open Passport Project

purposes, such as efficient cross-border interoperability, privacy-preserving verification using zero-knowledge proofs, and scalable management of credential revocation. Hence, commitments aren't strictly required to establish authenticity in these trusted scenarios but rather support an efficient, privacy-conscious, and interoperable credential verification framework

- **Off-Chain Data & Merkle Proofs:** The actual raw data that commitments refer to (passport details, names, birthdates, biometric info, educational degrees) are kept off-chain, likely in the user's digital wallet or in a secure repository controlled by the user. When the user needs to prove something, they generate a Merkle proof, using Merkle trees, that their specific credential is included under the on-chain root. For example, suppose the on-chain root corresponds to a tree that includes (leaf = hash(passport#12345, ownerDID)). The user's wallet will store the passport#12345 and its hash and the path in the Merkle tree. To prove to a verifier that their passport is in the registry, the wallet provides the Merkle path (a list of sibling hashes up to the root) and a ZK proof that they know a valid leaf. In many cases, the Merkle proof itself can be part of the ZK circuit – the circuit can take the leaf data and path as private inputs and publicly assert the root matches the known on-chain root, thus proving inclusion without even revealing which leaf it is (except whatever is implied by the proof statement). This combination of off-chain data with on-chain anchors achieves minimal disclosure (only hashes on-chain) and scalability (on-chain data is minimal, avoiding large storage of full credentials).

- **Selective Disclosure and Attribute Commitments:** Often a credential contains multiple attributes (think of an eID card containing name, birthdate, address, etc.). Instead of hashing them all together, it is possible to create an attribute-level commitment scheme, such as a Merkle tree where each leaf is a different attribute commitment. The holder can then reveal or prove one attribute at a time. For example, a diploma credential might commit to degree in separate leaves. If a verifier only needs to know the degree, the user proves inclusion of that leaf and that the root corresponds to an issuer-signed root Using Merkle trees and ZK, naturally supports this fine-grained disclosure: store as little on-chain as possible (ideally one commitment covering all attributes), and let the user's ZK proof selectively demonstrate the required attribute's truth.[8]

- **Revocation Data:** On-chain storage is very useful for revocation status because verifiers need an up-to-date, reliable source to check if an attestation is still valid. Common approaches include on-chain revocation registries (each credential or each issuer has a list or a bitmap of revoked credentials) or off-chain status lists with hashes anchored on-chain. For instance, if a user's credential is revoked, the issuer could update a "revocation Merkle tree" on-chain and the user's proof circuit would have to prove their credential's ID is not in the revocation tree. This can be done with a Merkle exclusion proof or by querying a status smart contract. EBSI is exploring status lists and blockchain anchoring for revocation[9], which we can integrate so that any credential that's been revoked is recognized across the network.

- **Storage of Nullifiers:** One specific piece of data it is often store on-chain in ZK systems is the nullifier. Nullifiers prevent double use of the same proof or credential. An on-chain contract can hold a mapping of nullifier values that have been seen. When a proof is verified, its nullifier is

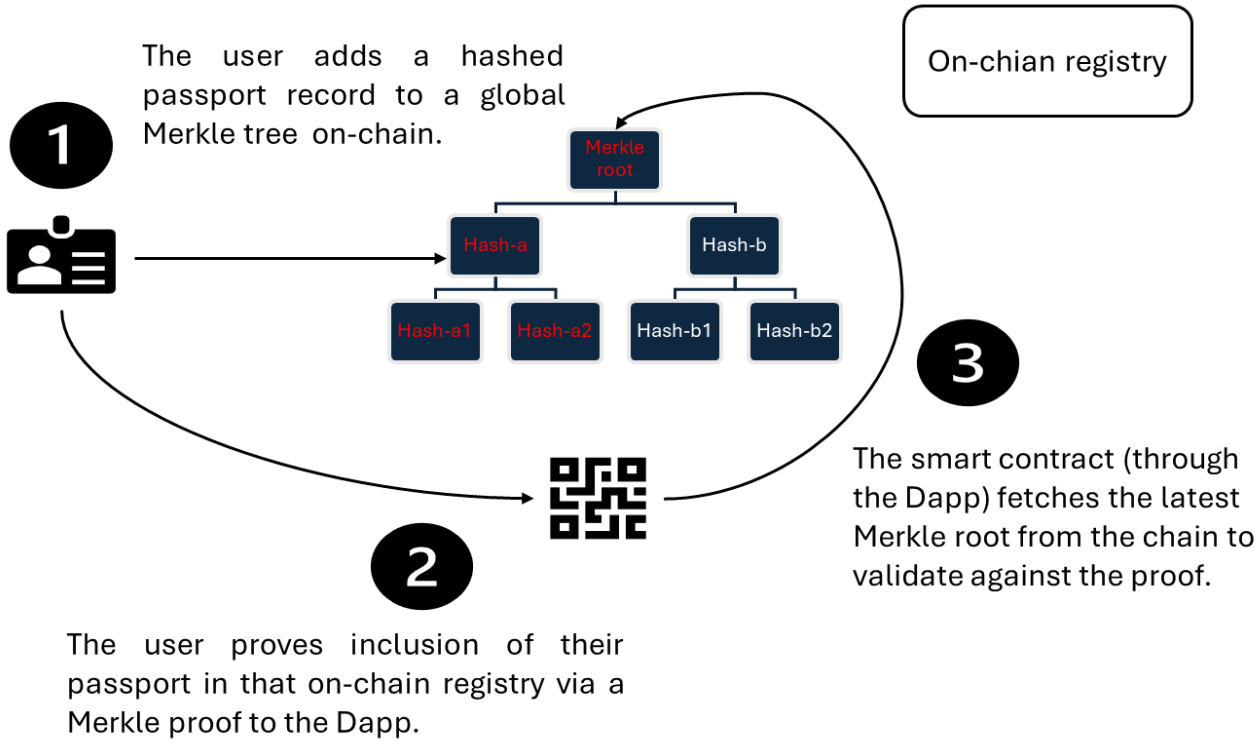
[8] How On-Chain Attestations Unlock Blockchain's Most Valuable Use Cases - Decrypt

[9] Securing Documents with Blockchain VCs | BCdiploma

recorded a small hash record. This on-chain memory of used nullifiers stops replay attacks globally. Nullifiers are typically random or unlinkable to the user's real ID, so storing them on-chain does not harm personal data identification strongly. The Ethereum Semaphore protocol[10], for example, uses on-chain nullifier storage in exactly this way for anonymous voting. Although not universally applicable to every verification scenario, nullifiers explicitly address common real-world scenarios such as single-use credentials scenarios involving uniqueness constraints and replay attack protection within identity verification.

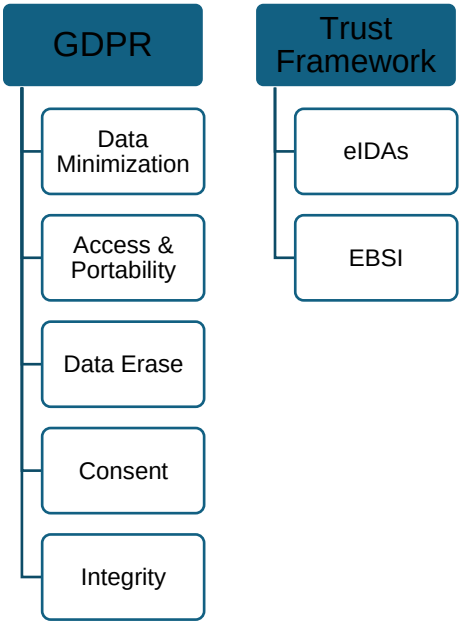
Example – Web3 Passport Storage Flow

- The user (or issuer) adds a hashed passport record to a global Merkle tree on-chain;
- The user proves inclusion of their passport in that on-chain registry via a Merkle proof to the Dapp;
- The ZK verifier contract fetches the latest Merkle root from the chain to validate against the proof.



[10] Semaphore Protocol - website

Regulatory Considerations



Data Minimization and Privacy via ZK Attestations

Data minimization is a core principle of the GDPR and is at the heart of our use of zero-knowledge proofs. Instead of the traditional approach where a user hands over an entire identity document (and thus overshares data) just to prove one fact, ZK attestations allow the user to prove only the required fact and nothing more. This cryptographic minimal disclosure enforces privacy by design.

For example, in a Web3 Passport scenario, if a user needs to prove they are over 18 to access a service, under a ZK system they present only a proof of the statement "age ≥ 18", certified by their government's attestation, without revealing their actual birth date or. The verifier learns no personal details – not even the exact age – only that the proof is valid. This dramatically reduces personal data exchanged, satisfying GDPR Article 5(1)(c) (data minimization) and Article 5(1)(b) (purpose limitation) because data provided is limited to what is strictly necessary for the purpose (age verification). It also mitigates risks of data misuse: a verifier cannot accidentally leak what it never saw in the first place.

Moreover, the user stays in full control of their information. There is no centralized database that collects all these attributes; instead, the credentials reside with the user, in their wallet app. With approaches, the user consents to each disclosure. ZK proofs elevate this by allowing the user to consent to share proof of X instead of actual data X. This aligns with GDPR's notion of data protection by design and by default (Article 25), as the system is engineered to technically prevent unnecessary data exposure.

Additionally, because on-chain data is limited to pseudonymous commitments and proofs, the risk of personal data breaches from the blockchain is minimized. Even if someone scans the entire EBSI ledger, they would find only hashes and perhaps public keys, but no plain personal info to link to an individual. To bolster this, techniques like the "privacy network effect" where hashes of passport data are hard to link to specific identifiers. By using a single large Merkle tree for all users, any given hash is lost in a crowd; as more users join, the difficulty of guessing which hash corresponds to which person

increases. This means that even an entity with access to some government cannot easily re-identify a user from the on-chain registry, because the space of possibilities and the mixing of contexts provide plausible deniability.

In addition to that, GDPR classifies hashed data as personal data if it can be reversed or linked to an individual. This design mitigates at the most any linkability by using salts or including user-specific randomizers in hashes where appropriate, and by relying on ZK proofs rather than putting even hashed attributes on-chain individually. For instance, a proof that "passport is valid and belongs to someone over 18" might reveal a nullifier and a yes/no answer, but not the passport number or holder's identity.

Each ZK proof generation can be seen as an act initiated by the user (or their agent, the wallet) with their consent. Unlike traditional KYC where a user might hand a document and lose control over how it's copied or stored, here the verifier gets a transient proof valid only for that moment (perhaps even with a one-time nullifier to prevent reuse). After verification, the verifier cannot further process the user's data beyond the allowed purpose, because they literally do not have the data – an elegant technical enforcement of purpose limitation.

In summary, ZK attestations operationalize the principle of minimal disclosure = minimal risk. By proving statements about data rather than sharing data, they enhance privacy far beyond what organizational policy alone can achieve. This not only aids GDPR compliance but also fosters user trust in digital identity systems, users may be more likely to adopt an EU Digital Wallet if they know it won't broadcast their personal info everywhere.

Compliance with eIDAS 2.0 and European Digital Identity Frameworks

The eIDAS 2.0 regulation introduces a framework for a European Digital Identity (EUDI) that includes European Digital Identity Wallets and cross-border recognition of electronic identification and electronic attestations of attributes (EAA). The system should be designed to fulfill the technical and legal criteria eIDAS 2.0 sets forth:

- **Use of Qualified or Trusted Issuers**: In the Web3 Passport, the issuers of credentials (e.g. a national government issuing a "citizenship" credential, or a health authority issuing a "vaccination certificate") would be recorded in a Trusted Issuers Registry on EBSI. [11] The verifiers (relying parties) will trust credentials because they trust the issuer's status. For instance, a verifier can check that the issuer's DID is listed as a qualified authority for that attribute type (e.g., only a government DID can issue a "nationality" credential). EBSI's governance ensured by Europeum EDIC, will determine that only legitimate bodies are in that list, solving the trust on first use problem.
- **Electronic Seals/Signatures:** eIDAS requires that any official electronic attestation is signed or sealed by the issuer to guarantee authenticity and integrity. In this design, every credential, the data container that is later proven via ZK, is digitally signed by the issuer. The zero-knowledge proof does not bypass this; rather, the proof circuit verifies the issuer's digital signature internally. For example, if a government issues a credential saying "DOB = Jan 1, 2000" signed with its private key, the ZK proof that "age >= 18" will include a check that this signature

[11] This is exactly what EBSI provides for – did:ebsi identities can be associated with a Legal Entity and verified against an official registry Request and verify Verifiable Credentials | EBSI hub.

is valid (without revealing DOB). This means the verifier is effectively getting a guarantee equivalent to checking the signature themselves. As long as the issuer's public key is trusted, the verifier can accept the proof as if the data was directly signed by the issuer. In fact, one could say the ZK proof itself is an "evidence token" derived from a valid credential.

- **Wallet Certification and Security:** This approach seems fitting into the reference EUDI Wallet architecture. The wallet would manage DIDs, store credentials, and crucially, generate presentations. In the reference, a "verifiable presentation" is often an extracted subset of data or a derived proof. We simply make that presentation a ZK proof instead of plaintext data. The wallet software would undergo conformity assessment to ensure, for example, it properly checks verifier requests and doesn't leak data. The wallet can enforce that only the requested attribute (and nothing more) is proven, matching the "selective disclosure" policies recommended by EBSI.
- **Interoperability and Formats:** The Web3 Passport should be able Open standards like JSON-LD, JWT, or SD-JWT for credentials. A ZK proof can be packaged in a Verifiable Presentation object (the W3C Data Model allows embedding a proof as an element). There are emerging standards to encode ZKPs in credentials. Ensuring the proof and credential formats are standardized will allow any conformant verifier implementation from any country to verify the proof given the proper libraries

Also is important to mention that The European Telecommunications Standards Institute (ETSI) Technical Report 119 476, focuses on elaborating a standardized approach for Zero Knoelwdge cryptography and may represent a critical milestone in the standardization of privacy-preserving cryptographic mechanisms, with a particular emphasis on Zero-Knowledge Proofs (ZKPs). This technical report is designed to assess and define how ZKPs can enhance trust services, identity management, and electronic signatures within the framework of the eIDAS Regulation.

Given the growing demand for self-sovereign identity (SSI) and verifiable credentials (VCs), ETSI TR 119 476 establishes guidelines on how ZK cryptographic primitives can be applied to ensure compliance with EU legal and regulatory requirements while maintaining the highest levels of privacy. This effort recognizes ZKPs as a foundational technology for verifiable digital identities by allowing individuals to prove specific attributes (such as age, citizenship, or professional qualifications) without revealing unnecessary personal data. The ETSI TR 119 476 outlines security and interoperability requirements for ZK systems and evaluates their compatibility with existing cryptographic frameworks used in European identity and trust services[12].

A major innovation within the ARF is its support for privacy-enhancing technologies, including Zero-Knowledge Proofs (ZKPs) and other advanced cryptographic techniques. The goal is to enable selective disclosure and minimal data exposure while ensuring that transactions involving digital identity remain legally recognized and auditable.

---

[12] ETSI TR 119476 Analysis of selective disclosure and zero-knowledge proofs applied to Electronic Attestation of Attributes, Work Programme, https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=67975&curItemNr=64&totalNrItems=213&optDisplay=100000&qSORT=TB&qETSI_ALL=&SearchPage=TRUE&qINCLUDE_SUB_TB=&qINCLUDE_MOVED_ON=&qEND_CURRENT_STATUS_CODE=11+WI%3BM58&qSTOP_FLG=N&qKEYWORD_BOOLEAN=&qCLUSTER_BOOLEAN=&qCLUSTER=17&qFREQUENCIES_BOOLEAN=&qSTOPPING_OUTDATED=&butExpertSearch=Search&includeNonActiveTB=FALSE&includeSubProjectCode=&qREPORT_TYPE=TUBE

The integration of ZKPs into the EUDI Wallet is driven by the need to strike a balance between user control over personal data and compliance with regulatory mandates. Through ZKPs, users can authenticate themselves to service providers without revealing excessive information, thereby mitigating risks associated with identity theft, unauthorized data collection, and third-party surveillance[13].

User rights Personal Data Protection

In terms of security, lawful processing, and user rights, the web3 Passport should be generated through applications that allow:

- **Lawful Basis and Consent:** Typically, the use of a digital identity credential for verification will be under the user's consent or a legal obligation on the verifier's side. In a Web3 Passport scenario, the user explicitly consents to share a proof when they tap "Approve" in their wallet for a verifier's request (this could be seen as consent under GDPR Article 6(1)(a)). Because the wallet intermediates every disclosure, there's no silent or automatic data collection – the user is in the loop. Moreover, since only derived data (like "Yes, over 18") is shared, the user's consent is very specific and limited in scope, as GDPR requires. If a legal obligation is the basis (say an online casino must verify age under law), the system still ensures no excess data flows beyond what the law demands, which aligns with GDPR's requirement that even legally required processing should not be excessive.
- **Right of Access and Portability:** A user has the right to access their personal data and port it. In this design, the user actually holds their data in the wallet, so they inherently have access to it at all times. They can view their credentials (like "My Web3 Passport contains: DOB, nationality, etc."). Portability (Art.20) is facilitated because credentials are based on open standards – a user could export a credential from one wallet app to another, or even present it outside of the blockchain context if needed (since the credential is fundamentally a signed data blob that could be verified independently of the ZK system). The use of DIDs and VCs means the identity data isn't locked into one processor's proprietary system; it's under the user's control and in an interoperable format.
- **Right to Erasure (Right to be Forgotten):** This right is tricky in blockchain contexts because data on-chain is immutable. However, our approach keeps personal data off-chain; the on-chain data (hashes, commitments) in isolation typically cannot identify the user without additional information. According to guidance by EU regulators, if data on-chain is truly anonymous then it's not personal data and erasure rights don't apply to it. Even if considering the on-chain commitment as personal data, the user can effectively "erase" their presence by revoking credentials and not using that identity further. For instance, if a user wants to revoke their Web3 Passport, the issuer can flag their credential as revoked (update the on-chain state). While the historical transaction might remain on-chain, it's just a random hash that is no longer active or usable. In practice, for GDPR compliance, one could also design the commitments to be updateable or replaceable such that the original commitment is no longer referenced anywhere once the user deletes their account. Additionally, the user can delete data from their own wallet (which is the main storage of PII). The blockchain never had their PII, and any verifiers never received it either, so the surface for "forgetting" is small. The biggest thing to consider is

[13] See from ARF: https://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework/blob/main/docs/discussion-topics/g-zero-knowledge-proof.md

audit logs: if a verifier logs that "User X (pseudonym) proved age on date Y", that log might be considered personal data (especially if pseudonym can be tied back). A privacy-respecting verifier should avoid storing even that, or if needed, store only ephemeral or necessary records. Since proofs are typically not re-useable (nullifiers prevent that), a verifier doesn't need to keep much.

- **Data Security and Integrity:** GDPR Article 32 requires controllers and processors to ensure appropriate security, including encryption. ZK proofs and commitments inherently use strong cryptography to protect data. Communications between wallet and verifier can be encrypted (e.g., using DIDComm or HTTPS). The verifiers and issuers never handle raw personal data beyond what they might already legally have (issuers obviously have it during issuance, but they're governmental and have their own GDPR obligations; verifiers ideally get none). The use of TEEs as discussed adds an extra layer where needed, which would be viewed favorably from a GDPR security standpoint. Data integrity is guaranteed by digital signatures and on-chain immutability – credentials cannot be tampered with undetectably, nor can someone forge a proof without the genuine credential.
- **Transparency to the User:** GDPR emphasizes that individuals should know what's happening with their data. In this system, the processes are relatively transparent to the user via the wallet's UI: the wallet can show "Verifier X is requesting to know if you are over 18. It will not receive any other information." The user can then approve or deny. That message itself is a transparent notice. Compare this to traditional systems where a user might not fully realize that showing an ID card also reveals their address and ID number – here, the wallet explicitly limits and communicates the data exchange.
- **Purpose Limitation and Further Processing:** Because the verifier only gets the answer to their specific query, they are technically incapable of using the data for any other purpose. This is a strong guarantee that the data won't be repurposed (unlike if they got a full ID, they might be tempted to copy the address or other details for marketing etc.). So the system inherently enforces purpose limitation – each proof is generated for a specific purpose and often can be tagged with that purpose (sometimes ZK proofs include a mechanism to bind them to a specific verifier or context to prevent reuse elsewhere). For example, it could be possible to incorporate the domain or request ID into the proof (so-called domain separation), so the proof can't be forwarded to another service. Ethereum's EIP-4361 ("Sign-in with Ethereum") and related proposals sometimes include a clause to prevent replay across domains. Similarly, proofs can include an audience restriction, ensuring they're valid only for the intended relying party. This addresses GDPR's concerns about data being shared beyond what the user agreed to.

Trust Models for Issuance and Verification

A digital identity system must define **who is trusted to issue credentials** and **how verifiers can trust what they see**. In a potential European Web3 Passport, the trust model leverages institutional trust (governments, authorities) combined with technical trust (blockchain consensus, cryptography):

Issuance Trust: Identities and attribute attestations aren't self-asserted; they come from authoritative sources. Under eIDAS, each member state will notify certain Identity Providers or Attribute Providers. For example, a country's interior ministry might be the issuer of citizenship credentials, universities

issue diploma credentials, etc. These issuers are trusted because they are legally accountable and usually certified. In the Web3 Passport system, each such issuer has a cryptographic identity (DID) linked to its real-world identity, and a public key on record. When it issues a credential, it signs it. So the root of trust for any given attestation is the issuer's signing key – just like in physical documents the root of trust is the government's seal.

To make this scalable, EBSI runs a Trusted Issuer Registry smart contract (or an off-chain governed list) where each issuer's DID and key are registered along with the types of credentials they can issue. Member states govern this list, therefore, any wallet or verifier can automatically validate the issuer: they check that the signature on a credential comes from a key that is listed for that credential type. If yes, and if the proof verifies, they can trust the content.

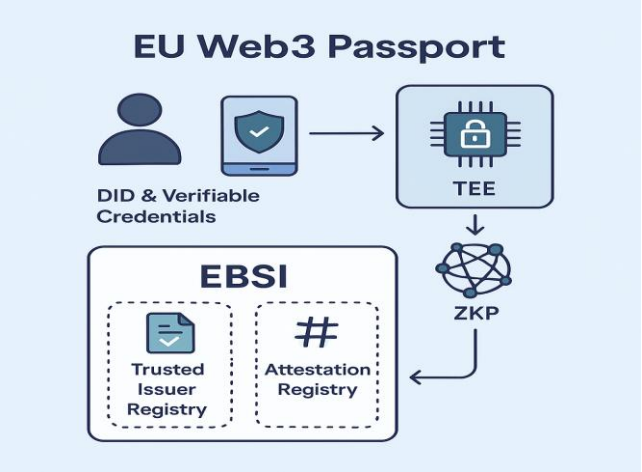
User Identity Binding: One challenge is ensuring the credential presented via ZK indeed belongs to the person presenting it. In a typical VC flow, a holder might prove control of the private key corresponding to the DID that the credential was issued to. With ZK technology it is possible to incorporate a similar check: e.g., the credential could contain the user's public key (or DID), and the ZK proof circuit ensures the prover knows the corresponding private key. This is often done by having the user sign something with their DID key and verifying that signature inside the proof. By doing so, we maintain the trust that the presenter is the rightful holder of the credential. Without this, someone could try to use another's credential if they somehow got hold of it. So the trust model at verification includes: trust the issuer and trust that the prover is the subject of the credential.

Verification Process: For a verifier (say a service provider in a country), trusting a proof involves trusting the following: (1) the cryptographic soundness of the ZK proof (which is a matter of trusting the ZK protocol and the implementation, (2) the issuer's authority (ensured by the signature and issuer registry), and (3) the integrity of the on-chain data (ensured by blockchain consensus). Europeum EDIC who run EBSI's consortium of nodes provides a high assurance that the on-chain records (like the Merkle root, issuer lists are tamper-proof and globally consistent. This is important: verifiers in all countries see the same state of the world. If an issuer revoked a credential and updated the Merkle root, all verifiers will eventually see that update on EBSI and proofs using the old state will fail.

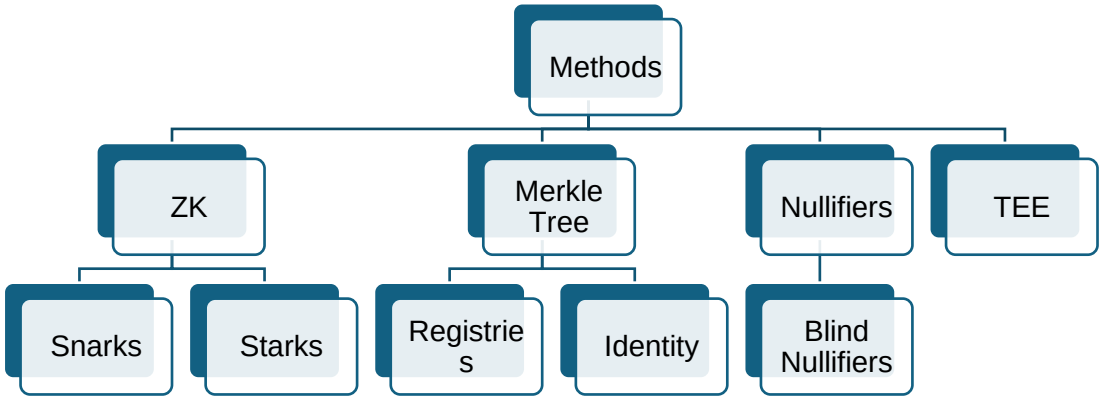
Role of Smart Contracts in Trust: The smart contracts (or ledger functions) act as impartial verifiers of proofs and enforce rules like nullifier one-time use. For on-chain verification use-cases the contract will only accept a proof if all checks pass (signature was valid, nullifier not seen, etc.), and then perhaps emit an event "Proof accepted for user X". If any step is wrong, the contract rejects. This removes human error or subjective decision at verification time. The trust is thus in the cryptographic protocol and the audited code. By open-sourcing the circuits and software, we ensure transparency.

TEE in Trust Model: TEEs can reduce the need to trust the operator's intentions, but the user must trust the hardware vendor and attestation. In a government context, it's conceivable that an EU-approved service could run TEEs for heavy tasks, but to keep the system maximally trustless, so it would be preferable to have TEEs as an enhancement rather than a necessity. The trust model can be tiered: a proof that was generated with a fully local process (or a transparent multi-party process) might be considered "Trustless Level 1", whereas a proof that involved a TEE might be "Trustless Level 2" where the verifier also checks an attestation from the TEE. The good news is that eIDAS and EU policy are generally open to hardware security modules), so a TEE is not foreign it could even be formally certified. But in this design, the aim is that verifiers do not have to trust any single party's infrastructure – only the issuer and the technology.

In summary, the trust model is a three-way handshake: Issuers trust the user with a credential, users trust issuers and the wallet, and verifiers trust issuers and the proof mechanism. The blockchain protocol (EBSI) and cryptography can mediate this trust by ensuring everyone sees the same truth and no one has to blindly trust anyone's word – they can verify mathematically. This is a more robust and transparent trust model than today's federated logins or paper document checks, and it fits within the legal trust fabric defined by eIDAS which provides the assurance that the issuers are who they claim to be and are accountable



References Cryptographic Methods



zk-SNARKs vs. zk-STARKs for Identity Proofs

Zero-Knowledge Proofs (ZKPs) come in various constructions. The leading choices for on-chain verification today are zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge) and zk-STARKs (Zero-Knowledge Scalable Transparent Arguments of Knowledge). Both allow a prover to convince a verifier of a statement's truth without revealing why it's true, but they differ in key ways that impact our implementation:

- **zk-SNARKs:** These proofs are extremely succinct (often just a few hundred bytes) and very fast to verify (a few milliseconds), which is ideal for blockchain use where efficiency is paramount. Classic SNARK systems (like Groth16, used in Zcash and many identity systems like iden3) do require a **trusted setup** – a one-time ceremony to generate public parameters that, if compromised, could allow false proofs. Modern SNARK variants like PLONK or Marlin have universal setups (which can be reused for many circuits) or even updatable setups, making the ceremony less of a vulnerability. SNARKs use heavy algebraic assumptions (elliptic curve pairings, etc.) and have relatively small prover computation compared to STARKs for typical statement sizes. For a Web3 Passport, SNARKs are attractive because verifying a proof on-chain (say in a smart contract on EBSI) is cheap and the proofs won't bloat transactions. Many existing open libraries (eg. Iden3, Semaphore, etc.) have opted for SNARKs for these reasons. We could imagine a setup ceremony at the European level (with many parties contributing randomness) to generate parameters for our circuits (e.g. for proving an issuer signature and an attribute range). Given the EU's collaborative nature, this is feasible. After that, every proof (of age, citizenship, etc.) could be verified with those parameters.
- **zk-STARKs:** These are a newer breed that avoids the need for a trusted setup by relying on collision-resistant hash functions and error-correcting codes. STARKs also have the advantage of **post-quantum security** (they're based on hashes, so quantum computers won't break them, whereas SNARKs on certain curves would be vulnerable). However, STARK proofs are larger (tens of kilobytes often) and take longer to verify. They also have high memory usage when proving. In an identity scenario, a STARK might be used to prioritize transparency and post-quantum resilience over performance. For instance, a "national ID STARK proof" could be generated by the wallet and verified either off-chain by the verifier's device or on a chain with high throughput. EBSI nodes might handle a few kilobytes proof, but it's less efficient. That said, as technology improves, STARK sizes are decreasing and hardware is improving.

Given current tech, SNARKs are likely the pragmatic choice, considering many reference circuits exist (for verifying signatures, Merkle proofs, etc.). For example, iden3's circom is an open library includes circuits for validating an ECDSA signature or a Pedersen hash inside a proof, which could be adapted. SNARKs like Groth16 yield ~128-byte proofs, which are easy to store or transmit, and their verification can be done in a smart contract with a few pairing operations. We could verify such proofs on EBSI easily.

One trade-off: SNARKs typically have a proving time that grows with circuit complexity, and some identity proofs can become complex. STARKs or specialized proofs might do better for that specific case. In a hybrid approach, one might use a SNARK for most things and a STARK for specific parts (or use Recursive proof composition – where a SNARK verifies a STARK or vice versa – but that's advanced).

For clarity: the "typical" identity ZK proof in our context might include these checks in the circuit: signature verification (by issuer), Merkle proof verification (to ensure credential is in issuer's state tree or user's state), attribute checks (like age >= 18), nullifier computation, and perhaps user's ownership signature.

Each of these has been implemented in SNARK circuits by different projects:

- Signature check: e.g., BLS or ECDSA verification can be done (iden3 uses BabyJubjub signatures which are SNARK-friendly).

- Merkle proof: SNARKs handle Merkle paths easily (a sequence of hash operations).
- Comparison (>=18): an integer comparison or range proof is straightforward in a circuit.
- Hashing (for nullifier): SNARKs can do Poseidon or MiMC hashing efficiently (these are ZK-friendly hash functions). So, SNARKs fit well with these tasks. STARKs would also handle them but possibly with larger overhead.

To conclude, for a government-backed system, a concern might be the toxic waste from SNARK setup. A mitigation for this may be by a large multi-party ceremony set up, the larger and more transparent, the safer to include all the Member States joining Europeum-EDIC.

A possible solution could be, use a well-vetted SNARK system (Groth16 or PLONK) for the initial implementation. Keep an eye on emerging transparent proofs. Possibly upgrade later to something like Marlin or Spartan that could allow some pre-processing trust assumptions. In any case, both SNARKs and STARKs uphold the zero-knowledge property and computational soundness needed. – the difference is mainly in practicality. For now, assuming considering the industry adoption I tend to assume SNARKs with trusted setup may be a preferable choice.

Merkle Trees and Merkle-Tree-Based Credentials

Merkle trees are a fundamental data structure for efficient verification of set membership and are heavily used in decentralized identity for scaling and privacy. In this architecture, Merkle trees appear in multiple ways:

- **Global Registry Trees:** As noted, one big Merkle tree can hold all issued attestations or at least references to them. For example, the web3 passport registry may use a sparse Merkle tree where each leaf corresponds to a passport hash entry. The advantage of a Merkle tree is that the root (32 bytes) is a succinct representation of millions of entries. Without Merkle trees, either store each credential separately on-chain, would be impractical and heavy, or require direct issuer online verification each time which breaks decentralization and user-controlled privacy. So Merkle trees allow verifiable off-chain storage – the data can be off-chain, but its integrity is anchored on-chain.
- **User Identity Trees:** Specific solution such as Iden3's identity protocol implements each identity as a Merkle tree of claims[14]. This means a user can accumulate claims from various issuers under one identity. The leaves might be things like "Claim: issuer A says attribute X = Y (with issuer A's signature)". The user then updates their identity state root on-chain whenever a new claim is added. This approach compartmentalizes each user's data. A verifier then needs to know the user's latest root (hence the state contract mapping of user IDs to roots)[15] and the proof from the user that a certain claim is in their tree and is valid. One benefit here is claiming revocation: if the user wants to remove a claim (maybe an outdated attestation or one they no longer want to present), they can update their tree and the old claim is no longer in the latest root, so it won't verify in proofs using the latest state. The on-chain state contract can even

[14] Iden3 protocol specs - Iden3 Documentation
[15] Configuration | Privado ID Documentation

enforce that proofs must use the most recent root to avoid someone using an old root that still had a now-revoked claim.

A Merkle proof is a sequence of hashes from a leaf to the root. Verifying it in a circuit means iteratively hashing the leaf with each sibling hash in the path, according to whether the leaf is the left or right child at that level, until you compute the root, then checking it matches the expected root (provided as a public input to the circuit, which in turn is tied to an on-chain value). Circuits are quite efficient at this if the hash function is SNARK-friendly. Industry standard solutions would likely use Poseidon or MiMC hash in circuits (these are designed to be efficient within SNARKs). If the on-chain commitments use a different hash (say SHA-256 because that's standard for passports), one approach is to either have a circuit implementation of SHA (which is doable but heavier) or double-store commitments (issuer provides a Poseidon hash for ZK and maybe a SHA-256 for their own reference).

In summary, Merkle trees give us a swiss-army knife: they enable compact representation, efficient proof of inclusion or exclusion, and structured commitments. They pair extremely well with ZK proofs, since a lot of logic can be reduced to "prove that this hash chain connects to that root". Our system will lean heavily on Merkle proofs to avoid large data transfers and to allow verifiers to rely on a common consistent state (the root).

Nullifiers to Prevent Replay and Ensure Uniqueness

Nullifiers are unique tokens derived from a secret that a prover reveals when they use a credential or identity in a ZK proof, in order to mark that use and prevent it from being reused. The concept has been used into identity services to prevent double-signing or double-claiming while preserving anonymity. In a potential Web3 Passport design, nullifiers serve a few purposes:

- **One-Person-One-Action Constraints:** Suppose a scenario of an token airdrop: each eligible user (say EU residents) can claim tokens once. We don't want someone with the same credential to claim multiple times. By including a nullifier in the proof, which is computed as a function of the user's identity and the context (in this case the airdrop event), we can have a smart contract check and reject duplicate nullifiers. The nullifier could be Hash(user_id, event_id), where user_id is some internal representative of that user in the circuit (for example, the index of their leaf in the registry tree, or the hash of their passport number). The proof would output this hash as a public output, and the contract would store it once. If another claim comes with the same nullifier, it's a double spend and gets rejected.
- **Unlikability Across Verifications:** Ideally, each verification a user does should not be linkable to another, unless the user wants that. Nullifiers help in that linking them per verifier or per use-case. For instance, a user proving "I'm human" to two different dApps could use two different nullifiers: Hash(user_id, dApp1) and Hash(user_id, dApp2). If dApp1 and dApp2 compare notes, they see different random-looking nullifiers, so they cannot tell it's the same user behind both proofs. However, within dApp1, if the user tries to register twice, they'd present the same nullifier both times (since the context dApp1 is the same), and the contract or server would notice and prevent the second registration. This provides Sybil resistance without sacrificing anonymity across contexts. Each relying party gets a unique pseudonym for the user (the nullifier), which is consistent for that party so they know it's the same entity each time, but different across parties so there's no global tracking.

- **Preventing Replay of the Same Proof:** Even outside of Sybil scenarios, a nullifier can include a nonce or session so that a proof cannot be copied and replayed later by someone else. For example, the verifier could send a challenge which then influences the nullifier. If an eavesdropper somehow got the proof, they couldn't use it again because the verifier would expect a different challenge next time. This is more of an anti-replay protection for protocol security.

When designing nullifiers, it's crucial to ensure they don't compromise user privacy. A naive approach, such as using hash(passport_number), creates a unique but easily linkable identifier. If someone associates this hash with an individual, all their actions could be traced. Worse, a government (or any entity with access to passport databases) could directly link on-chain activity to a person if the passport hash is used as a nullifier.

Using a static nullifier (e.g., hash(passport_number)) allows the issuer to track users by monitoring on-chain interactions. This breaks privacy since any entity with access to passport records could deanonymize users[16].

To enhance privacy, nullifiers should include additional randomness or context:

- **Blinded Nullifiers**: Instead of using a fixed value, introduce a secret or dynamic factor, such as:\text{nullifier} = H(\text{government_signature}, \text{user_secret}, \text{context})
- **Decentralized Approach**: Instead of relying on a central server to sign attestations, the user can add their own secret (e.g., their DID private key).
- **Randomized Credential Identifiers**: Instead of directly using a passport number, the government could issue a credential containing a randomly generated unique ID

This approach ensures that each nullifier is used only once per context. A key application of this in Web3 Passports is Proof of Uniqueness, which verifies that "I am a unique human." To achieve this, a nullifier is generated as an output. If different dApps ask, "Is this a unique human?" the way nullifiers are structured determines privacy.

If a global nullifier is used, the same identifier appears across all applications, which can compromise cross-app anonymity. Alternatively, a system like Privado ID[17] Proof of Uniqueness assigns a different nullifier per application, preserving privacy across services. For global uniqueness, the nullifier can be tied to an identity and a universal context, such as context = "uniqueness", in the context of Europe this can be tied to EBSI Trust issuers registry. This functions as a pseudonymous ID that ensures no two users generate the same nullifier, preventing duplicate registrations. However, this also means that across all applications using the same "uniqueness" context, the same nullifier appears, making it possible to track an individual across services. To balance privacy and functionality, our system could allow both per-service nullifiers (ensuring privacy across apps) and an optional global unique nullifier for cases where users consent to proving they are not receiving duplicate benefits. For instance, if multiple banks want to prevent someone from applying for loans under different identities, they could agree on a shared context for credit checks. However, such use cases require careful consideration to avoid privacy risks and unintended tracking.

---

[16] Privacy preserving nullifiers for proof of identity applications - zk-s[nt]arks - Ethereum Research

[17] https://docs.privado.id/docs/verifier/query-builder/

If nullifier is based on a specific credential or identity, a person with multiple passports (dual citizenship) could circumvent one-per-person rules, or someone could even fraudulently obtain a second credential. This is a broader identity federation problem – nullifiers can enforce "one per credential" but not necessarily "one per natural person" if people have multiple credentials. The European approach could mitigate that and try to issue singular identities for EBSI credentials for wallet per person, linked to one root eID. It's also possible to use biometrics (like face/fingerprint) to ensure uniqueness, but that raises privacy issues and is outside our current scope, except to note that if truly needed, one could incorporate a privacy-preserving biometric check. Possibly in the future, ZK + secure biometrics (like zk-SNARK proving you have registered an iris in a database exactly once) could complement this. But for now, a solid trust model assumes each person has at most one valid credential representing Web3 Passport at a time, and issuers/policies will limit issuing multiple to the same person.

Nullifiers are thus an essential piece to enforce constraints without revealing identity. They effectively allow anonymous accountability – you can be anonymous, but you can't cheat because your actions are linkable in the specific ways that matter (like double spending).

TEE-Enhanced zkProof Computation

Combining TEEs with ZK can yield the best of both worlds: ZK ensures privacy of the data in transit, and TEEs ensure security of the data and code in use. How TEEs can enhance the cryptographic aspects and performance of ZK proofs:

- **Accelerating Proof Generation:** ZK proof generation (proving) can be very resource-intensive (CPU and memory heavy). In scenarios where a user device is too slow, a TEE on a server or cloud can accelerate this by using more powerful hardware while still keeping data safe. For instance, an enclave could utilize vectorized instructions, ample memory, or even GPU (if a secure GPU enclave exists) to produce a proof in seconds that would take a smartphone minute or be impossible due to memory. The TEE's guarantee, a more effective user inclusion engagement, at the same time it means the user can trust that the server didn't copy their private data during the process. This is a form of delegated proving. It doesn't make the proof any different for the verifier (the proof is still the same format), but it moves the workload. This could be crucial for proofs that involve large signatures or many attributes.
- **Secure Key Management for ZK:** ZK proofs often require the prover to have certain secrets (like the randomness used in a signature, or the blinding factor in a commitment). TEEs can store these secrets and only allow their use in correct ways. For example, imagine each user has a secret user_secret that must never be revealed but is needed to compute nullifiers. The user could load this into their phone's secure enclave; the ZK proving software queries the enclave to get H(user_secret, context) for the nullifier, without ever seeing user_secret itself. This way, even if the phone OS is compromised, the attacker cannot produce a valid proof on behalf of the user because they can't get the necessary secret or compute the nullifier properly. This blends well with ZK since ZK can hide values, but if malware controls the device, it could simply ask the user's ZK app to produce proofs it shouldn't (like for a fraudulent context). With a TEE enforcing policies (e.g., only allow proof generation when user consents via secure UI, etc.), it strengthens the overall system.

- **Multi-Party Computation (MPC) with TEEs:** Another advanced idea: multiple TEEs could jointly compute a proof or a piece of data such that no single one has full knowledge (some companies do "MPC in SGX" for managing keys). In identity, not directly needed, but could be used for something like distributed issuance or linking data from multiple sources without revealing them to each other, then producing a combined attestation.

Concluding Thoughts

A potential Web3 Passport is built on the core concept of the OASIS, Baseline Protocol[18]. It shares key cryptographic features that ensure privacy, integrity, and verifiability in decentralized systems. Both leverage zero-knowledge verification to enable trustless proof validation without exposing underlying data, and they utilize on-chain commitments such as cryptographic hashes and Merkle roots to ensure data integrity while keeping sensitive information off-chain. These foundational elements align both protocols with modern cryptographic best practices for secure and privacy-preserving transactions.

However, the Web3 Passport extends these common features with a standardized identity framework tailored for Web3 applications. Unlike the Baseline Protocol, which primarily focuses on enterprise process synchronization and business data integrity, the Web3 Passport is designed for decentralized identity verification and cross-border regulatory compliance. It achieves this by integrating Decentralized Identifiers (DIDs)—such as did:ebsi, which ensures interoperability within the European identity ecosystem—and Verifiable Credentials (VCs), which serve as structured, cryptographically signed data containers that users can selectively disclose.

A significant distinction in the Web3 Passport's architecture is its use of Merkleized Credentials, where identity attestations are structured within a Merkle tree. This allows users to prove possession of verified credentials while minimizing on-chain data exposure.

The combination of on-chain commitment anchoring, Merkle proof verification, and zero-knowledge attestations ensures that identity verification remains trustless, privacy-preserving, and scalable across Web3 ecosystems[19]. By aligning with European frameworks such as eIDAS 2.0 and the European Blockchain Services Infrastructure (EBSI), the Web3 Passport establishes a privacy-first approach to digital identity that bridges the gap between legal recognition and Web3 interoperability.

The solution also offers unique features such as the combination with Truste Execution Environment (TEE). TEEs complement Zero Knowledge by ensuring the integrity of the data computation side, whereas ZK handles the data compression and minimization side. This duality is powerful: one can imagine a future European identity stack where citizen data is always handled either under the protection of strong cryptography (ZK, encryption) or strong hardware isolation (TEE), or both.

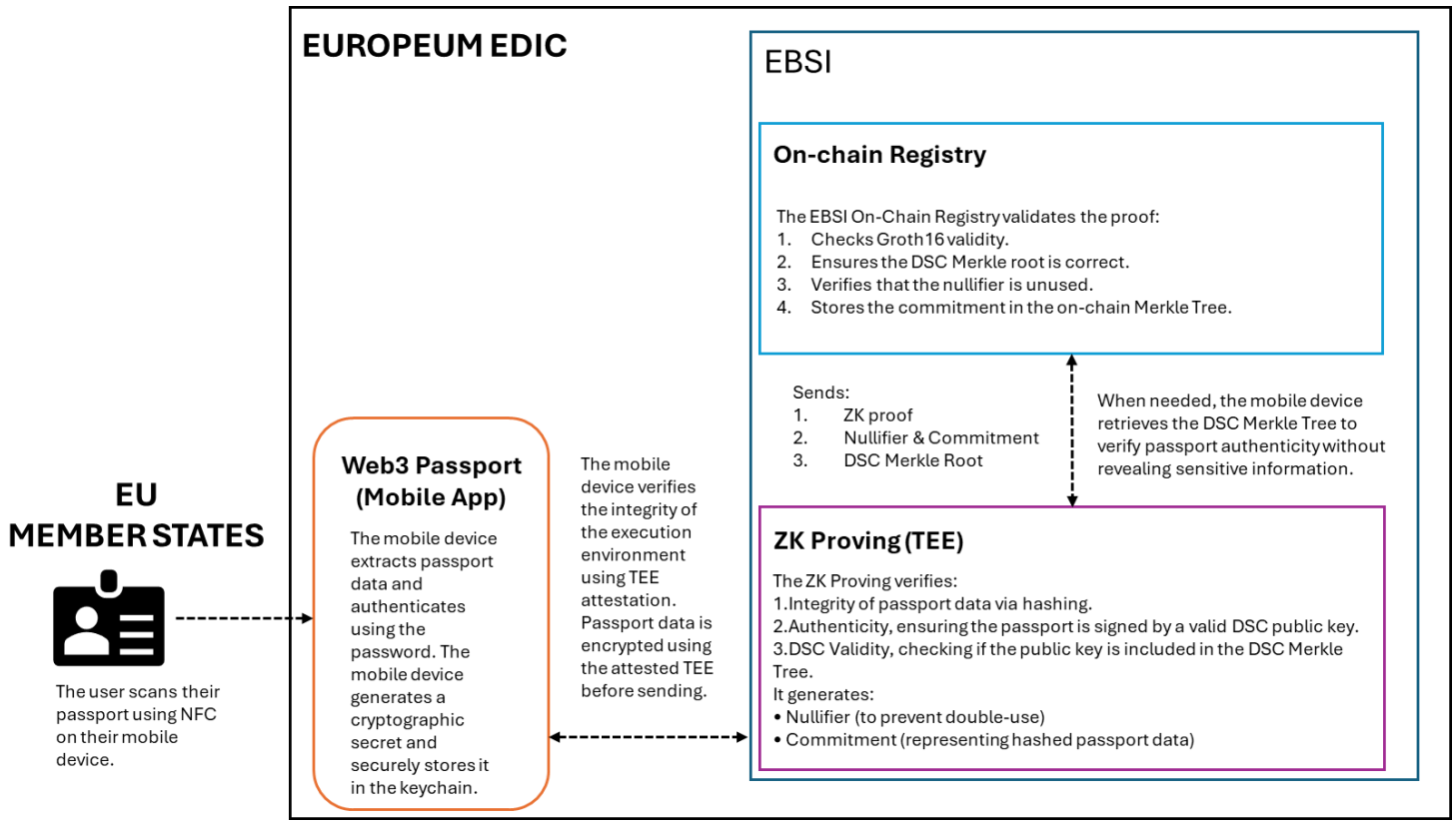
---

[18] Baseline Protocol - Website

[19] Ultimately, while both protocols share a foundation in zero-knowledge cryptography and on-chain verifiability, the Web3 Passport's adoption of standardized digital identity models (DIDs, VCs, and Merkleized Credentials) makes it uniquely suited for decentralized and regulatory-compliant identity verification.

A Web3 Passport could similarly use TEEs to ensure that, say, an online verification service cannot leak any data or be tricked into outputting a false proof (because the enclave won't sign off unless everything checks out), while the ZK proof ensures the service never even learns the data to leak in the first place.

Furthermore, standardizing access to Web3 services through a blockchain-based on-chain identifier is crucial to fostering secure, privacy-preserving authentication. Leveraging industry practices, zero-knowledge proofs, TEEs, and the EBSI registry ensures a scalable, interoperable, and regulatory-compliant model for decentralized identity verification. As the European digital identity landscape evolves, a unified approach to on-chain attestations will be key to harmonizing cross-border interactions and advancing the vision of a user-centric, decentralized internet.



Users and passport holders issued by Member States extract data using NFC, which is then turned into a credential following a specific credential schema and stored into an application (mobile or web). Then the user credential represents a standardized data format that undergoes to merklization process and subsequently zero-knowledge verification to determine the correctness of the computation (over the passport data) in a minimized way and its uniqueness and determine the inclusion into the registry (EBSI).

The Europeum European Digital Infrastructure Consortium (EDIC) on EBSI presents an opportunity to standardize the use of blockchain-based identity solutions, bridging traditional digital services with

Web3 applications. By leveraging on-chain identifiers, ZK-proof attestations, and TEE-secured verification, the Web3 Passport can streamline user authentication across different digital environments. This standardization benefits government services, financial institutions, and decentralized applications alike. I try to share a potential system architecture design describing how the process would look like.

Potential business cases include DeFi onboarding for investors, where verified credentials can ensure regulatory compliance while preserving privacy. In the following deliverables I will try to provide concrete examples of use case that leverage sources of verifies data to establish on chain attestations that can be applied in industrial scenarios. Specifically, I will explore the use on chain attestation to establish data ownership of a specific metadata that frame Intellectual property rights or governance voting rights. In the first scenario the on chain may establish trust to empower more efficient NFT licensing service for IP owners, then allowing creators to attach on chain attestations to their intellectual property, ensuring provenance and licensing rights enforcement on-chain. On the second scenario, members of associations, cooperatives, consortium, could benefit from on chain voting leveraging tamper-proof, identity-linked attestations, ensuring fair execution of their token power and governance operations without exposing voters' personal data. These applications underscore the need for a unified, privacy-first identity standard that supports interoperable Web3 services within the European regulatory framework.

Eugenio Reggianini